ALGORITHM ENGINEERING FOR COVERING PROBLEMS IN HYPERGRAPHS

Synopsis

Submitted in the partial fulfillment for the requirement of

Doctor of Philosophy

In

Mathematics

Submitted by

Pushkar Singh

Under the supervision of

Prof. Gur Saran
Department of Mathematics

Department of Mathematics, Faculty of Science
Dayalbagh Educational Institute (Deemed University)
Dayalbagh, Agra-282005

2022

1. Introduction

Many important computational problems are difficult to solve optimally [4]. In fact, many of those problems are NP-hard, which means that no polynomial-time algorithm exists that solves the problem optimally for all input instances unless $P = NP$. Some of the famous NP hard optimization problems are Covering problems (Partial vertex cover problem, Set multicover problem) [13,15,18]. If $P \neq NP$, it is unlikely to find an algorithm simultaneously having 1) an optimal solution 2) in polynomial time 3) at any instance. At least one of these requirements must be relaxed to solve a NP-hard optimization problem. One of the ways to tackle is to give up on the requirement of solving the problem optimally, and to settle for a solution which is close to optimal. Over the years, several techniques have been adopted to guarantee a nearby close optimal solution [35].

Heuristic approach [8] includes backtrack search and its variants, mathematical programming methods, local search, genetic algorithms, tabu search etc. Some of these methods are guaranteed to find an optimal solution, though they may take exponential time; others are guaranteed to run in polynomial time, though they may not return an optimal solution. Therefore, using a heuristic approach makes no guarantee as to their effectiveness [4].

To deal with these problems, two other approaches are commonly adopted, i.e., Approximation algorithms and Randomized algorithms. Approximation algorithms [34,35] aim to find solutions whose costs are as close to be optimal solution as possible in polynomial time whereas Randomized algorithms [29] can be looked as to designing algorithms giving optimal solutions in expected polynomial time. Many different techniques are used in these kind of algorithms which can be classified into a) linear programming based algorithms b) semi definite programming based algorithms c) randomized and derandomized algorithm etc. Within each class, standard algorithm design techniques such as greedy method, dynamic programming, primal dual methods, dual fitting, metric embedding, linear and semi definite programming relaxations etc. are often adopted [30]. It turns out that approximation algorithm guarantees on the quality of the solution found to be within a multiplicative factor of an optimal solution [8].

In recent years, many areas of theoretical computer science have shown growing interest in solving problems arising in real-world applications, experiencing a shift to more application-motivated

research. However, for many decades, researchers have been mostly using mathematical methods for analyzing and predicting the behavior of algorithms. The new demand for algorithms that are of practical utility has now raised the need to refine and reinforce the traditional theoretical approach with experimental studies, fitting the general models and techniques used by theoreticians to actual existing machines, and bringing algorithmic research back to its roots. The whole process of designing, analyzing, implementing, tuning, debugging and experimentally evaluating algorithms is usually referred to as Algorithm Engineering [11]. Algorithm Engineering views algorithmics also as an engineering discipline rather than a purely mathematical discipline. A major goal of Algorithm Engineering is to define standard methodologies and realistic computational models for the analysis of algorithms.

2. Approximation algorithm for covering problems

Definition 1 [28]: An optimization problem π is characterized by three components:

- [Instances] D: a set of input instances.
- [Solutions] $S(I)$: the set of all feasible solutions for an instance $I \in D$.
- [Cost/Value] f: a function which assigns a cost/value to each solution, i.e., $f: S(I) \rightarrow \mathbb{R}^+$.

Each solution $s \in S$ is associated with a cost $c(s)$. An optimization problem π is called a minimization (maximization) problem if one wishes to find minimum (maximum) cost solutions respectively for every instance. An algorithm $\mathcal{A}$ for an optimization problem π, maps every instance I to a feasible solution $s \in S$. For each instance I of a problem, let $OPT(I)$ denote the value of an optimal solution to instance I.

Definition 2 [4]: A ρ-approximation algorithm for an optimization problem is a polynomial time algorithm that for every instance I of the problem produces a solution whose value is within a factor of ρ of $OPT(I)$. Equivalently, we say that $\mathcal{A}$ is an approximation algorithm with an approximation ratio ρ.

An algorithm $\mathcal{A}$ for a minimization problem is called a ρ-approximation algorithm, for some $\rho \geq 1$, if the algorithm produces for any input I a solution whose value $\mathcal{A}(I) \leq \rho . OPT(I)$. An algorithm $\mathcal{A}$ for

a maximization problem is called a ρ-approximation algorithm, for some $\rho \leq 1$, if the algorithm produces for any input I a solution whose value $\mathcal{A}(I) \geq \rho . OPT(I)$.

Problem 1 (Vertex Cover) [34]: A vertex cover for an undirected graph $G = (V, E)$ is a set $V' \subseteq V$ such that if (u, v) is an edge of G then either $u \in V'$ or $v \in V'$. The size of vertex cover is the number of vertices in V'. The vertex cover problem is to find a vertex cover of minimum size in a given undirected graph.

A weighted vertex cover is defined as follows: Given $G = (V, E)$ and weight function $w: V \to \mathbb{Q}^+$, find a vertex cover of minimum total weight. A feasible solution is a set $S \subseteq V$ that is a vertex cover for G. Its objective function value is the sum of weights of all vertices in S. A feasible solution of minimum weight is an optimal solution.

Many traditional heuristics for vertex cover are known. Perhaps the most obvious is the greedy algorithm, in which the vertex of maximum remaining degree (incident on the maximum number of edges) is repeatedly removed from the graph and added to the cover until all edges are covered. The algorithm actually performs poorly on many classes of graphs and has no fixed performance bounds. An alternative algorithm is focused on finding a maximum matching in the graph: while edges remain, choose an arbitrary edge, add both endpoints to the cover, and remove both vertices from the graph. The algorithm has a fixed performance bound of 2, i.e., the resulting cover is at most twice the optimum cover.

Definition 3 [5]: A hypergraph is a generalization of graph in which an edge can join any number of vertices. A hypergraph $\mathcal{H}$ is a pair $\mathcal{H} = (V, \varepsilon)$ where V is a set of vertices and ε is a set of non-empty subsets of V called hyperedges or edges.

Problem 2 (Set Cover) [34]: Given a universe U of n elements, a collection of subsets of U, $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$, and a cost function $c: \mathcal{S} \to \mathbb{Q}^+$, a set cover is to find a minimum cost subcollection $\mathcal{S}'$ of $\mathcal{S}$ that covers all elements of U, i.e., each element in U appears in atleast one set $S_i \in \mathcal{S}'$.

The standard greedy algorithm in every run through the while loop, selects the set maximizing the ratio between the number of uncovered elements and the cost of the set. It then modifies the input sets to remove the newly covered elements from them, effectively removing some elements of U since

they have already been covered and it is irrelevant whether they are covered again or not. Therefore, the algorithm has a fixed performance bound of $\log d + \mathcal{O}(1)$ where d is maximum vertex degree in graph, i.e., the resulting cover is at most logarithmic times the optimum cover.

Definition 4 [7]: For a minimization problem, a α-approximation randomized algorithm returns a feasible solution S of expected cost $Exp[c(S)] \leq \alpha.OPT$. For a maximization problem, a α-approximation randomized algorithm returns a feasible solution S of expected cost $Exp[c(S)] \geq \frac{OPT}{\alpha}$.

Problem 3 (k-partial Vertex Cover) [13]: Given a hypergraph $\mathcal{H} = (V, \varepsilon)$ with a set of vertices $|V| = n$, and set of hyperedges $|\varepsilon| = m$. The problem is to find a minimum cardinality k-partial vertex cover in hypergraphs where a set $X \subseteq V$ is called a k-partial vertex cover for $\mathcal{H}$ if at least k edges of $\mathcal{H}$ are incident in X.

Problem 4 (Set b-multicover) [15]: Given a hypergraph $\mathcal{H} = (V, \varepsilon)$ with a set of vertices $|V| = n$, and set of hyperedges $|\varepsilon| = m$. The problem is to find a minimum cardinality set b-multicover in hypergraph where a set b-multicover in $\mathcal{H}$ is a set of edges $C \subseteq \varepsilon$ such that every vertex in V belongs to at least b edges in C. For $b = 1$ we have the well-known set cover problem.

The randomized approximation algorithm for set b-multicover in hypergraph works by firstly picking an empty set C, which extends to a feasible cover [15]. The problem is then solved using the LP-relaxation technique in polynomial time, with some known polynomial-time procedure, e.g. the interior point method. Next, the edges are removed with a corresponding LP-variable of value zero, because they will not be taken into the set b-multicover by randomized rounding. Then randomized rounding is performed followed by a greedy repairing step, if the solution is infeasible, so the resulting cover is always feasible.

3. Literature Review

The history of Covering problems (Vertex cover & Set cover) in graphs and hypergraphs is long and distinguished. The approximation complexity of these optimization problems has been studied for nearly 4 decades.

Lovasz[26] in 1975 showed that the ratio of the optimal integral and the fractional covers of a hypergraph does not exceed $1 + \log d$, where d is the maximum degree of a hypergraph. He further

confirmed that this theorem might replace the probabilistic method in certain circumstances. Chavtal[10] in 1979 proposed a greedy heuristic for the set covering problem in a hypergraph. He compared the heuristic solution to the true optimum solution and found that the ratio between the two grows at most logarithmically in the maximum vertex degree. The heuristic proposed yields a bound of 2 on the ratio between the heuristic solution and the optimal vertex cover solution. Hochbaum[22] in 1982 delivered $\mathcal{O}(n^3)$ step solution for the set covering problem in hypergraphs. Krivelevich[25] in 1997 considered a weighted set covering problem on l-uniform hypergraphs where he applied two different approaches : Firstly, ratio of the optimal integral and the fractional covers approach on maximal degree bounded by Δ and obtained an approximation ratio $l\left(\frac{1-c}{\frac{1}{\Delta^{l-1}}}\right)$. Secondly, a local ratio approach on general l-uniform hypergraph and obtained an approximation ratio $l\left(\frac{1-c}{n^{\frac{l-1}{l}}}\right)$. Semi local optimization, a powerful new technique was given by Duh and Furer[12] in 1997 to approximate a k-set cover. Halperin[20] in 2000 improved approximations for vertex cover problems on l-uniform hypergraphs with maximal degree at most Δ by achieving a ratio of $l-\big(1-o(1)\big)\,\frac{l(l-1)\ln\ln\Delta}{\ln\Delta}$ as Δ gets large, and for on l-uniform hypergraphs with n vertices by achieving a ratio of $l-\big(1-o(1)\big)\,\frac{l\ln\ln n}{\ln n}$ as n gets large.

While most of them investigated the set cover and hitting set problem for hypergraphs in polynomial time approximation, others aimed at the complexity of non-approximation. Lund[27] in 1993 proved that it is hard to compute efficiently good approximate solutions for minimization of set covering problem. He proved that the set covering cannot be approximated with ratio $c\,log\,n$ for any $c<\frac{1}{4}$ unless NP is contained in $DTIME[n^{poly\log n}]$. Further, Raz and Safra[32] in 1997 introduced a low degree test that uses the restriction of low degree polynomials to planes and showed the test to be of very small error probability which enabled to prove a low error characterization of NP in terms of PCP. One application of this characterization of NP approximated set cover problem to be within a logarithmic factor was NP-hard. Feige[16] in 1998 proved $\big(1-o(1)\big)\log n$ to be the threshold below

which set cover cannot be approximated efficiently, unless NP has super polynomial time algorithms. Alon et al.[1] in 2006 provided algorithmic construction of sets for k-restrictions.

It has been an open question on the improvement of 2 approximation for the k-partial vertex cover problem in graphs. Bar-Yehuda[3] in 2001 gave an algorithm based on local-ratio method for the k-partial vertex cover problem in hypergraphs with an approximation guarantee l, where l is the maximum edge size. Halperin[21] in 2002, improved the approximation for k-partial vertex cover problem in graphs from $\left(2-\Theta\left(\frac{1}{d}\right)\right)$ to $\left(2-\Theta\left(\frac{\ln\ln d}{\ln d}\right)\right)$ using semidefinite programming and randomized rounding. For graphs with maximum vertex degree at most Δ, Gandhi et al.[17] in 2004 gave the first algorithm with approximation ratio smaller than 2 using primal/dual approach. Khot and Regev[24] in 2008 presented unique games conjecture (UGC) for the hitting set problem in l-uniform hypergraph with constant l and approximated that a ratio better than l cannot be achieved in polynomial time. Since the hitting set problem in hypergraphs is a special case of k-vertex cover problem in hypergraphs with $k = number\ of\ hyperedges$, approximation hardness also holds for k-partial vertex cover problem in hypergraphs. El Ouali et al.[13] in 2014 proposed a randomized algorithm of hybrid type which combines LP-based randomized rounding and greedy repairing on k-partial vertex cover problem in hypergraphs. For $l \geq 3$ and constant D where D be the maximum edge degree of hypergraph, the algorithm achieves an approximation ratio of $l\left(1-\Omega\left(\frac{1}{D+1}\right)\right)$ which can be improved to $l\left(1-\Omega\left(\frac{1}{\Delta}\right)\right)$ if Δ is constant and $k \geq \frac{m}{4}$ where $m = number\ of\ hyperedges$. For l-uniform hypergraphs and $l \leq 4\Delta$ where l and Δ being constants, improved ratio of $l\left(1-\frac{l-1}{4\Delta}\right)$ was obtained. For a larger class of hypergraphs where $l \geq 3$ is not constant but D is constant they showed a ratio of $l\left(1-\frac{1}{6D}\right)$. Finally, for hypergraphs with non-constant l, but constant Δ and $k \geq \frac{m}{4}$ they obtained a ratio of $l\left(1-\frac{2-\sqrt{3}}{6\Delta}\right)$.

One of the famous generalized covering problem in combinatorial optimization is Set b-multicover problem for $b \geq 2$. Johnson[23] in 1974 and Vazirani[34] in 2003 provided a greedy algorithm where an approximation ratio of $1+\ln(l)$ can be obtained for constant l in $\mathcal{O}(nmb)$ time such that l is the

maximum edge size. Hall and Hochbaum[19] in 1986 achieved an approximation ratio of δ in $\mathcal{O}(m.max\{n,m\})$ where $\delta = \Delta - b + 1$. Peleg et al.[31] in 1997 was the first who approximated below δ with a ratio $\delta\left(1 - \left(\frac{c}{n}\right)^{\frac{1}{\delta}}\right)$ through randomized rounding, where $c > 0$ is a constant. Berman et al.[6] in 2007 improved the result through randomized algorithm that yields an approximation ratio of $\left(1 + o(1)\right)ln\left(\frac{l}{b-l}\right)$, when $\frac{l}{b}$ is large enough and an approximation ratio of $1 + 2\sqrt{\frac{l}{b}}$, if $\frac{l}{b}$ is small. Khot and Regev[24] in 2008 under unique games conjecture (UGC) stated that for $b = 1$ an approximation ratio of $\delta - \epsilon = \Delta - \epsilon$ cannot be achieved in polynomial time. El Ouali et al.[15] in 2016 proposed a randomized algorithm of hybrid type which combines LP-based randomized rounding and greedy repairing on Set b-multicover problem in hypergraphs. They achieved an approximation ratio of $\delta\left(1 - \frac{11(\Delta-b)}{72l}\right)$ for hypergraphs with maximum edge size $l \in \mathcal{O}\left(max\left\{(nb)^{\frac{1}{5}}, n^{\frac{1}{4}}\right\}\right)$. Recently in 2020, Gorgi et al.[18] presented a polynomial time deterministic threshold algorithm with conditioned randomized rounding steps. The algorithm yields an approximation ratio of max $\left\{\frac{148}{149}\delta, \left(1 - \frac{(b-1)e^{\frac{\delta}{4}}}{94l}\right)\delta\right\}$. This result improves the previous result since no restriction on parameter l is set. Moreover a polynomial time algorithm was presented with an approximation ratio of $\frac{5}{6}\delta$ for hypergraphs with $l \leq (1 + \epsilon)\bar{l}$ for any fixed $\epsilon \in \left[0, \frac{1}{2}\right]$, where $\bar{l}$ is the average edge size.

4. Algorithm Engineering

Formerly, in classical algorithmics, the analysis of algorithms for combinatorial problems focused on the theoretical analysis of asymptotic worst-case runtimes. As a consequence, the major aim was to develop asymptotically faster algorithms or to improve the existing ones. Over the years, several methods and techniques have been adopted to develop such sophisticated algorithms. However, asymptotically fast algorithms need not be efficient in practice. The algorithm may perform poorly on typical real-world instances, where alternative methods may be much faster, or the algorithm may be too complex to be implemented for a specific task. Therefore, for use in real-world applications,

robust algorithms are needed not only to offer good asymptotic performance but are also designed and experimentally evaluated to meet practical demands. In order to reduce gaps between theory and practice, a new research field of algorithm engineering copes with these kinds of problems. Algorithm engineering is simply sometimes referred as experimental algorithmics [11]. An important goal of algorithm engineering is to speed up the transfer of algorithmic knowledge into applications.

Algorithm engineering cycle [9] usually starts with some specific application (1) and tries to find a realistic model (2) for it such that the solutions obtained matches the requirements. The main cycle starts with an initial algorithmic design (3) on how to solve the model. Based on this design the algorithm is analyzed (4) from the theoretical point of view, e.g., to obtain performance guarantees (5) such as asymptotic runtime, approximation ratios, etc. The process continues with algorithmic development by implementing (6) the proposed algorithm. Usually this step marks as the most important step in algorithm engineering because not only theoretical ideas are transformed into actual code but also a constant checkup on the error and improvement of runtime analysis is performed. The final major piece in this cycle are the set of experiments (7) performed on real world data (8) input instances which allows us to compare different algorithms on a testing ground that is relevant for the practice. The aim of this cycle is to use the knowledge obtained during the steps of analysis, implementation, and experimentation to find improved algorithmic designs.

Algorithm engineering as a cycle of design, analysis, implementation, and experimental evaluation is driven by falsifiable hypotheses [33]. A final by-product of the algorithm engineering cycle is a kind of algorithm library (9), where it can be later used for further algorithmic development that builds upon the obtained results. Figure 1 illustrates the algorithm engineering cycle.

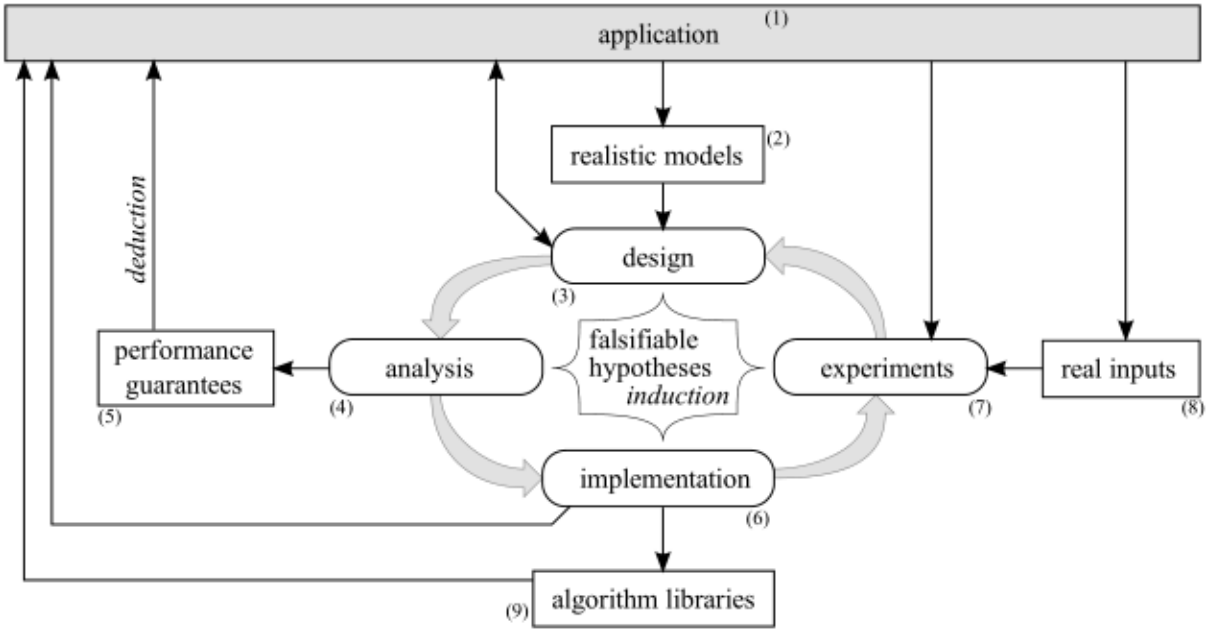

Figure 1: The algorithm engineering cycle[9]

5. Objective

It is proposed to study algorithm engineering for approximation algorithms for covering problems in hypergraphs, specifically as reported in the works of El Ouali et al.[13,15], as such a study has not been reported in literature surveyed so far, focusing on the following aspects:

- Generation of large graphs
- Implementation of algorithms
- Experiments and evaluation
- Derandomization of randomized approximation algorithms
- Implementation and test of the derandomized version
- Comparison of randomized and derandomized versions
- Improved randomized algorithm for set multicover and vertex cover in hypergraphs

6. References

[1] Alon N, Moshkovitz D, Safra S : Algorithmic construction of sets for k-restrictions. ACM Trans Algorithms 2, 153–177 (2006).

[2] Alon N, Spencer J : The probabilistic method, 2nd edn. Wiley Interscience, Hoboken, NJ (2000).

[3] Bar-Yehuda R.: Using homogeneous weights for approximating the partial cover problem. J. Algorithms 39(2), 137–144 (2001).

[4] Berg M de, “Lecture Notes on Advanced Algorithms (21L45)”, Dept. Comput. Sci., Eindhoven University of Technology (2013).

[5] Berge C : Hypergraphs-combinatorics of finite sets. North Holland Mathematical Library (1989).

[6] Berman P, DasGupta B, Sontag E : Randomized approximation algorithms for set multicover problems with applications to reverse engineering of protein and gene networks. Discrete Appl. Math.155(6–7), 733–749 (2007).

[7] Bhattacharyya A, “ Lecture notes on Approximation Algorithms”, E0 249 (2015).

[8] Chekuri C, “Lecture Notes on Approximation Algorithms : Introduction”, CS 583 (2018).

[9] Chimani M., Klein K. : Algorithm engineering: Concepts and practice. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.) Experimental Methods for the Analysis of Optimization Algorithms, pp. 131–158 (2010).

[10] Chvátal V : A greedy heuristic for the set covering problem. OMEGA 4(3), 233–235 (1979).

[11] Demetrescu C., Finocchi I., Italiano G.F., : Algorithm engineering. Bulletin ofthe EATCS 79 48–63 (2003).

[12] Duh R, Fürer M : Approximating k-set cover by semi-local optimization. In Proceedings of the 29th annual ACM symposium on theory of computing (STOC ’97), pp 256–264, May (1997).

[13] El Ouali M, Fohlin H, Srivastav A : An Approximation Algorithm for the Partial Vertex Cover Problem in Hypergraphs. J. Comb. Optim. (2014).

[14] El Ouali M, Fohlin H, Srivastav A : A randomised approximation algorithm for the hitting set problem. Theor. Comput. Sci. 555, 23–34 (2014).

[15] El Ouali M, Munstermann P, Srivastav A : Randomized Approximation for the Set Multicover Problem in Hypergraphs Algorithmica (2016) 74: 574.

[16] Feige U : A treshold of ln n for approximating set cover. J ACM 45(4), 634–652 (1998).

[17] Gandhi R, Khuller S, Srinivasan A : Approximation algorithms for partial covering problems. J Algorithms 53(1), 55–84 (2004).

[18] Gorgi A, El Ouali M, Srivastav A, Hachimi M : Approximation Algorithm for Multicovering problem (2020).

[19] Hall N.G., Hochbaum D.S. : A fast approximation algorithm for the multicovering problem. Discrete Appl. Math. 15, 35–40 (1986).

[20] Halperin E : Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. ACM-SIAM Symp Discret Algorithms 11, 329–337 (2000).

[21] Halperin E, Srinivasan A : Improved approximation algorithms for the partial vertex cover problem. In 5th international workshop on approximation algorithms for combinatorial optimization problems, LNCS, vol 2462, pp 161–174 (2002).

[22] Hochbaum D.S. : Approximation algorithms for the set covering and vertex cover problems. SIAM J Comput 11(3), 555–556 (1982).

[23] Johnson D.S. : Approximation algorithms for combinatorial problems. J.Comput.Syst.Sci.9, 256–278 (1974).

[24] Khot S, Regev O : Vertex cover might be hard to approximate to within 2-epsilon. J Comput Syst Sci 74(3), 335–349 (2008).

[25] Krivelevich J : Approximate set covering in uniform hypergraphs. J Algorithms 25(1), 118–143 (1997).

[26] Lovász L : On the ratio of optimal integral and fractional covers. Discrete Math 13, 383–390 (1975).

[27] Lund C, Yannakakis M : On the hardness of approximating minimization problems. J Assoc Comput Mach 41, 960–981 (1994).

[28] Motwani R, “Lecture Notes on Approximation Algorithms: Volume I,” Dept. Comput. Sci., Stanford Univ., Stanford, CA, Tech. Rep. CS-TR-92-1435 (1992).

[29] Motwani R, Raghavan P : Randomized algorithms. Cambridge University Press, Cambridge (1995).

[30] Ngo Hung Q, “Lecture Notes on Introduction to Approximation Algorithms”, CSE 531: Algorithm Analysis and Design (2004).

[31] Peleg D, Schechtman G, Wool A : Randomized approximation of bounded multicovering problems. Algorithmica 18(1), 44–66 (1997).

[32] Raz R, Safra S : A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In: Proceedings 29th ACM symposium on theory of computing, pp 475–484 (1997).

[33] Sanders P : Algorithm Engineering – An Attempt at a Definition. In Efficient Algorithms, volume 5760 of Lecture Notes in Computer Science, pages 321–340. Springer, (2009).

[34] Vazirani, V.V. : Approximation Algorithms. Springer, Berlin (2003).

[35] Williamson D.P. and Shmoys D.B., The design of approximation algorithms. Cambridge University Press (2011).